# NAG Toolbox

# nag_correg_quantile_linreg (g02qg)

## 1 Purpose

nag_correg_quantile_linreg (g02qg) performs a multiple linear quantile regression. Parameter estimates and, if required, confidence limits, covariance matrices and residuals are calculated. nag_correg_quantile_linreg (g02qg) may be used to perform a weighted quantile regression. A simplified interface for nag_correg_quantile_linreg (g02qg) is provided by nag_correg_quantile_linreg_easy (g02qf).

## 2 Syntax

```
[df, b, bl, bu, ch, res, state, info, ifail] = nag_correg_quantile_linreg
(sorder, intcpt, weight, dat, isx, y, tau, b, iopts, opts, state, 'n', n, 'm',
m, 'ip', ip, 'wt', wt, 'ntau', ntau)
[df, b, bl, bu, ch, res, state, info, ifail] = g02qg(sorder, intcpt, weight,
dat, isx, y, tau, b, iopts, opts, state, 'n', n, 'm', m, 'ip', ip, 'wt', wt,
'ntau', ntau)
```

## 3 Description

Given a vector of $n$ observed values, $y = \{y_i : i = 1, 2, \ldots, n\}$, an $n \times p$ design matrix $X$, a column vector, $x$, of length $p$ holding the $i$th row of $X$ and a quantile $\tau \in (0, 1)$, nag_correg_quantile_linreg (g02qg) estimates the $p$-element vector $\beta$ as the solution to

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \sum_{i=1}^{n} \rho_\tau \left( y_i - x_i^{\mathrm{T}} \beta \right) \tag{1}$$

where $\rho_\tau$ is the piecewise linear loss function $\rho_\tau(z) = z(\tau - I(z < 0))$, and $I(z < 0)$ is an indicator function taking the value 1 if $z < 0$ and 0 otherwise. Weights can be incorporated by replacing $X$ and $y$ with $WX$ and $Wy$ respectively, where $W$ is an $n \times n$ diagonal matrix. Observations with zero weights can either be included or excluded from the analysis; this is in contrast to least squares regression where such observations do not contribute to the objective function and are therefore always dropped.

nag_correg_quantile_linreg (g02qg) uses the interior point algorithm of Portnoy and Koenker (1997), described briefly in Section 11, to obtain the parameter estimates $\hat{\beta}$, for a given value of $\tau$.

Under the assumption of Normally distributed errors, Koenker (2005) shows that the limiting covariance matrix of $\hat{\beta} - \beta$ has the form

$$\Sigma = \frac{\tau(1 - \tau)}{n} {H_n}^{-1} J_n {H_n}^{-1}$$

where $J_n = n^{-1} \sum_{i=1}^{n} x_i x_i^{\mathrm{T}}$ and $H_n$ is a function of $\tau$, as described below. Given an estimate of the covariance matrix, $\hat{\Sigma}$, lower ($\hat{\beta}_L$) and upper ($\hat{\beta}_U$) limits for an $(100 \times \alpha)\%$ confidence interval can be calculated for each of the $p$ parameters, via

$$\hat{\beta}_{Li} = \hat{\beta}_i - t_{n-p,(1+\alpha)/2} \sqrt{\hat{\Sigma}_{ii}}, \hat{\beta}_{Ui} = \hat{\beta}_i + t_{n-p,(1+\alpha)/2} \sqrt{\hat{\Sigma}_{ii}}$$

where $t_{n-p,0.975}$ is the 97.5 percentile of the Student's $t$ distribution with $n - k$ degrees of freedom, where $k$ is the rank of the cross-product matrix $X^{\mathrm{T}} X$.

Four methods for estimating the covariance matrix, $\Sigma$, are available:

(i) Independent, identically distributed (IID) errors

Under an assumption of IID errors the asymptotic relationship for $\Sigma$ simplifies to

$$\Sigma = \frac{\tau(1-\tau)}{n}(s(\tau))^2\left(X^{\mathrm{T}}X\right)^{-1}$$

where $s$ is the sparsity function. nag_correg_quantile_linreg (g02qg) estimates $s(\tau)$ from the residuals, $r_i = y_i - x_i^{\mathrm{T}}\hat{\beta}$ and a bandwidth $h_n$.

(ii) Powell Sandwich

Powell (1991) suggested estimating the matrix $H_n$ by a kernel estimator of the form

$$\hat{H}_n = (nc_n)^{-1}\sum_{i=1}^{n} K\left(\frac{r_i}{c_n}\right)x_i x_i^{\mathrm{T}}$$

where $K$ is a kernel function and $c_n$ satisfies $\lim_{n\to\infty} c_n \to 0$ and $\lim_{n\to\infty}\sqrt{n}c_n \to \infty$. When the Powell method is chosen, nag_correg_quantile_linreg (g02qg) uses a Gaussian kernel (i.e., $K = \phi$) and sets

$$c_n = \min(\sigma_r, (q_{r3} - q_{r1})/1.34) \times \left(\Phi^{-1}(\tau + h_n) - \Phi^{-1}(\tau - h_n)\right)$$

where $h_n$ is a bandwidth, $\sigma_r, q_{r1}$ and $q_{r3}$ are, respectively, the standard deviation and the 25% and 75% quantiles for the residuals, $r_i$.

(iii) Hendricks–Koenker Sandwich

Koenker (2005) suggested estimating the matrix $H_n$ using

$$\hat{H}_n = n^{-1}\sum_{i=1}^{n}\left[\frac{2h_n}{x_i^{\mathrm{T}}\left(\hat{\beta}(\tau + h_n) - \hat{\beta}(\tau - h_n)\right)}\right]x_i x_i^{\mathrm{T}}$$

where $h_n$ is a bandwidth and $\hat{\beta}(\tau + h_n)$ denotes the parameter estimates obtained from a quantile regression using the $(\tau + h_n)$th quantile. Similarly with $\hat{\beta}(\tau - h_n)$.

(iv) Bootstrap

The last method uses bootstrapping to either estimate a covariance matrix or obtain confidence intervals for the parameter estimates directly. This method therefore does not assume Normally distributed errors. Samples of size $n$ are taken from the paired data $\{y_i, x_i\}$ (i.e., the independent and dependent variables are sampled together). A quantile regression is then fitted to each sample resulting in a series of bootstrap estimates for the model parameters, $\beta$. A covariance matrix can then be calculated directly from this series of values. Alternatively, confidence limits, $\hat{\beta}_L$ and $\hat{\beta}_U$, can be obtained directly from the $(1-\alpha)/2$ and $(1+\alpha)/2$ sample quantiles of the bootstrap estimates.

Further details of the algorithms used to calculate the covariance matrices can be found in Section 11.

All three asymptotic estimates of the covariance matrix require a bandwidth, $h_n$. Two alternative methods for determining this are provided:

(i) Sheather–Hall

$$h_n = \left(\frac{1.5\left(\Phi^{-1}(\alpha_b)\phi\left(\Phi^{-1}(\tau)\right)\right)^2}{n\left(2\Phi^{-1}(\tau) + 1\right)}\right)^{\frac{1}{3}}$$

for a user-supplied value $\alpha_b$,

(ii) Bofinger

$$h_n = \left(\frac{4.5\left(\phi\left(\Phi^{-1}(\tau)\right)\right)^4}{n\left(2\Phi^{-1}(\tau) + 1\right)^2}\right)^{\frac{1}{5}}$$

nag_correg_quantile_linreg (g02qg) allows optional arguments to be supplied via the **iopts** and **opts** arrays (see Section 12 for details of the available options). Prior to calling nag_correg_quantile_linreg (g02qg) the optional parameter arrays, **iopts** and **opts** must be initialized by calling nag_correg_optset (g02zk) with **optstr** set to **Initialize** = g02qg (see Section 12 for details on the available options). If bootstrap confidence limits are required (**Interval Method** = BOOTSTRAP XY) then one of the random number initialization functions nag_rand_init_repeat (g05kf) (for a repeatable analysis) or nag_rand_init_nonrepeat (g05kg) (for an unrepeatable analysis) must also have been previously called.

# 4 References

Koenker R (2005) *Quantile Regression* Econometric Society Monographs, Cambridge University Press, New York

Mehrotra S (1992) On the implementation of a primal-dual interior point method *SIAM J. Optim.* **2** 575–601

Nocedal J and Wright S J (1999) *Numerical Optimization* Springer Series in Operations Research, Springer, New York

Portnoy S and Koenker R (1997) The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute error estimators *Statistical Science* **4** 279–300

Powell J L (1991) Estimation of monotonic regression models under quantile restrictions *Nonparametric and Semiparametric Methods in Econometrics* Cambridge University Press, Cambridge

# 5 Parameters

## 5.1 Compulsory Input Parameters

1: **sorder** – INTEGER

Determines the storage order of variates supplied in **dat**.

*Constraint*: **sorder** = 1 or 2.

2: **intcpt** – CHARACTER(1)

Indicates whether an intercept will be included in the model. The intercept is included by adding a column of ones as the first column in the design matrix, $X$.

**intcpt** = 'Y'
An intercept will be included in the model.

**intcpt** = 'N'
An intercept will not be included in the model.

*Constraint*: **intcpt** = 'N' or 'Y'.

3: **weight** – CHARACTER(1)

Indicates if weights are to be used.

**weight** = 'W'
A weighted regression model is fitted to the data using weights supplied in array **wt**.

**weight** = 'U'
An unweighted regression model is fitted to the data and array **wt** is not referenced.

*Constraint*: **weight** = 'U' or 'W'.

4: **dat**$(lddat, :)$ – REAL (KIND=nag_wp) array

The first dimension, $lddat$, of the array **dat** must satisfy

if **sorder** = 1, $lddat \geq$ **n**;
otherwise $lddat \geq$ **m**.

The second dimension of the array **dat** must be at least **m** if **sorder** $= 1$ and at least **n** if **sorder** $= 2$.

The $i$th value for the $j$th variate, for $i = 1, 2, \ldots,$ **n** and $j = 1, 2, \ldots,$ **m**, must be supplied in

**dat**$(i, j)$ if **sorder** $= 1$, and

**dat**$(j, i)$ if **sorder** $= 2$.

The design matrix $X$ is constructed from **dat**, **isx** and **intcpt**.

5: **isx**(**m**) – INTEGER array

Indicates which independent variables are to be included in the model.

**isx**$(j) = 0$
The $j$th variate, supplied in **dat**, is not included in the regression model.

**isx**$(j) = 1$
The $j$th variate, supplied in **dat**, is included in the regression model.

*Constraints*:

**isx**$(j) = 0$ or 1, for $j = 1, 2, \ldots,$ **m**;
if **intcpt** = 'Y', exactly **ip** $- 1$ values of **isx** must be set to 1;
if **intcpt** = 'N', exactly **ip** values of **isx** must be set to 1.

6: **y**(**n**) – REAL (KIND=nag_wp) array

$y$, the observations on the dependent variable.

7: **tau**(**ntau**) – REAL (KIND=nag_wp) array

The vector of quantiles of interest. A separate model is fitted to each quantile.

*Constraint*: $\sqrt{\epsilon} <$ **tau**$(j) < 1 - \sqrt{\epsilon}$ where $\epsilon$ is the ***machine precision*** returned by nag_machine_precision (x02aj), for $j = 1, 2, \ldots,$ **ntau**.

8: **b**(**ip**, **ntau**) – REAL (KIND=nag_wp) array

If **Calculate Initial Values** = NO, **b**$(i, l)$ must hold an initial estimates for $\hat{\beta}_i$, for $i = 1, 2, \ldots,$ **ip** and $l = 1, 2, \ldots,$ **ntau**. If **Calculate Initial Values** = YES, **b** need not be set.

9: **iopts**(:) – INTEGER array

**Note**: the dimension of this array is dictated by the requirements of associated functions that must have been previously called. This array **must** be the same array passed as argument **iopts** in the previous call to nag_correg_optset (g02zk).

Optional parameter array, as initialized by a call to nag_correg_optset (g02zk).

10: **opts**(:) – REAL (KIND=nag_wp) array

**Note**: the dimension of this array is dictated by the requirements of associated functions that must have been previously called. This array **must** be the same array passed as argument **opts** in the previous call to nag_correg_optset (g02zk).

Optional parameter array, as initialized by a call to nag_correg_optset (g02zk).

11: **state**(:) – INTEGER array

**Note**: the actual argument supplied **must** be the array **state** supplied to the initialization routines nag_rand_init_repeat (g05kf) or nag_rand_init_nonrepeat (g05kg).

If **Interval Method** = BOOTSTRAP XY, **state** contains information about the selected random number generator. Otherwise **state** is not referenced.

## 5.2 Optional Input Parameters

1: **n** – INTEGER

*Default*: the dimension of the array **y** and the first dimension of the array **dat**. (An error is raised if these dimensions are not equal.)

The total number of observations in the dataset. If no weights are supplied, or no zero weights are supplied or observations with zero weights are included in the model then $\mathbf{n} = n$. Otherwise $\mathbf{n} = n+$ the number of observations with zero weights.

*Constraint*: $\mathbf{n} \geq 2$.

2: **m** – INTEGER

*Default*: the dimension of the array **isx** and the first dimension of the array **dat**. (An error is raised if these dimensions are not equal.)

$m$, the total number of variates in the dataset.

*Constraint*: $\mathbf{m} \geq 0$.

3: **ip** – INTEGER

*Default*: the first dimension of the array **b**.

$p$, the number of independent variables in the model, including the intercept, see **intcpt**, if present.

*Constraints*:

$1 \leq \mathbf{ip} < \mathbf{n}$;
if **intcpt** = 'Y', $1 \leq \mathbf{ip} \leq \mathbf{m} + 1$;
if **intcpt** = 'N', $1 \leq \mathbf{ip} \leq \mathbf{m}$.

4: **wt**(:) – REAL (KIND=nag_wp) array

The dimension of the array **wt** must be at least **n** if **weight** = 'W'

If **weight** = 'W', **wt** must contain the diagonal elements of the weight matrix $W$. Otherwise **wt** is not referenced.

When

**Drop Zero Weights** = YES
If $\mathbf{wt}(i) = 0.0$, the $i$th observation is not included in the model, in which case the effective number of observations, $n$, is the number of observations with nonzero weights. If **Return Residuals** = YES, the values of **res** will be set to zero for observations with zero weights.

**Drop Zero Weights** = NO
All observations are included in the model and the effective number of observations is **n**, i.e., $n = \mathbf{n}$.

*Constraints*:

If **weight** = 'W', $\mathbf{wt}(i) \geq 0.0$, for $i = 1, 2, \ldots, \mathbf{n}$;
The effective number of observations $\geq 2$.

5: **ntau** – INTEGER

*Default*: the dimension of the array **tau** and the second dimension of the array **b**. (An error is raised if these dimensions are not equal.)

The number of quantiles of interest.

*Constraint*: $\mathbf{ntau} \geq 1$.

## 5.3 Output Parameters

1: **df** – REAL (KIND=nag_wp)

The degrees of freedom given by $n-k$, where $n$ is the effective number of observations and $k$ is the rank of the cross-product matrix $X^{\mathrm{T}}X$.

2: **b**(**ip**, **ntau**) – REAL (KIND=nag_wp) array

**b**$(i,l)$, for $i = 1,2,\ldots,$ **ip**, contains the estimates of the parameters of the regression model, $\hat{\beta}$, estimated for $\tau =$ **tau**$(l)$.

If **intcpt** = 'Y', **b**$(1,l)$ will contain the estimate corresponding to the intercept and **b**$(i+1,l)$ will contain the coefficient of the $j$th variate contained in **dat**, where **isx**$(j)$ is the $i$th nonzero value in the array **isx**.

If **intcpt** = 'N', **b**$(i,l)$ will contain the coefficient of the $j$th variate contained in **dat**, where **isx**$(j)$ is the $i$th nonzero value in the array **isx**.

3: **bl**(**ip**, :) – REAL (KIND=nag_wp) array

The second dimension of the array **bl** will be **ntau** if **Interval Method** $\neq$ NONE.

If **Interval Method** $\neq$ NONE, **bl**$(i,l)$ contains the lower limit of an $(100\times\alpha)\%$ confidence interval for **b**$(i,l)$, for $i = 1,2,\ldots,$ **ip** and $l = 1,2,\ldots,$ **ntau**.

If **Interval Method** = NONE, **bl** is not referenced.

The method used for calculating the interval is controlled by the optional parameters **Interval Method** and **Bootstrap Interval Method**. The size of the interval, $\alpha$, is controlled by the optional parameter **Significance Level**.

4: **bu**(**ip**, :) – REAL (KIND=nag_wp) array

The second dimension of the array **bu** will be **ntau** if **Interval Method** $\neq$ NONE.

If **Interval Method** $\neq$ NONE, **bu**$(i,l)$ contains the upper limit of an $(100\times\alpha)\%$ confidence interval for **b**$(i,l)$, for $i = 1,2,\ldots,$ **ip** and $l = 1,2,\ldots,$ **ntau**.

If **Interval Method** = NONE, **bu** is not referenced.

The method used for calculating the interval is controlled by the optional parameters **Interval Method** and **Bootstrap Interval Method**. The size of the interval, $\alpha$ is controlled by the optional parameter **Significance Level**.

5: **ch**(**ip**, **ip**, :) – REAL (KIND=nag_wp) array

The last dimension of the array **ch** will be **ntau** if **Interval Method** $\neq$ NONE and **Matrix Returned** = COVARIANCE and at least **ntau** + 1 if **Interval Method** $\neq$ NONE, IID or BOOTSTRAP XY and **Matrix Returned** = H INVERSE

Depending on the supplied optional parameters, **ch** will either not be referenced, hold an estimate of the upper triangular part of the covariance matrix, $\Sigma$, or an estimate of the upper triangular parts of $nJ_n$ and $n^{-1}H_n^{-1}$.

If **Interval Method** = NONE or **Matrix Returned** = NONE, **ch** is not referenced.

If **Interval Method** = BOOTSTRAP XY or IID and **Matrix Returned** = H INVERSE, **ch** is not referenced.

Otherwise, for $i,j = 1,2,\ldots,$ **ip**, $j \geq i$ and $l = 1,2,\ldots,$ **ntau**:

If **Matrix Returned** = COVARIANCE, **ch**$(i,j,l)$ holds an estimate of the covariance between **b**$(i,l)$ and **b**$(j,l)$.

If **Matrix Returned** = H INVERSE, **ch**$(i,j,1)$ holds an estimate of the $(i,j)$th element of $nJ_n$ and **ch**$(i,j,l+1)$ holds an estimate of the $(i,j)$th element of $n^{-1}H_n^{-1}$, for $\tau =$ **tau**$(l)$.

The method used for calculating $\Sigma$ and $H_n^{-1}$ is controlled by the optional parameter **Interval Method**.

6: **res**(**n**, :) – REAL (KIND=nag_wp) array

The second dimension of the array **res** will be **ntau** if **Return Residuals** = YES.

If **Return Residuals** = YES, **res**$(i, l)$ holds the (weighted) residuals, $r_i$, for $\tau =$ **tau**$(l)$, for $i = 1, 2, \ldots,$ **n** and $l = 1, 2, \ldots,$ **ntau**.

If **weight** = 'W' and **Drop Zero Weights** = YES, the value of **res** will be set to zero for observations with zero weights.

If **Return Residuals** = NO, **res** is not referenced.

7: **state**(:) – INTEGER array

8: **info**(**ntau**) – INTEGER array

**info**$(i)$ holds additional information concerning the model fitting and confidence limit calculations when $\tau =$ **tau**$(i)$.

| Code | Warning |
|---|---|
| 0 | Model fitted and confidence limits (if requested) calculated successfully |
| 1 | The function did not converge. The returned values are based on the estimate at the last iteration. Try increasing **Iteration Limit** whilst calculating the parameter estimates or relaxing the definition of convergence by increasing **Tolerance**. |
| 2 | A singular matrix was encountered during the optimization. The model was not fitted for this value of $\tau$. |
| 4 | Some truncation occurred whilst calculating the confidence limits for this value of $\tau$. See Section 11 for details. The returned upper and lower limits may be narrower than specified. |
| 8 | The function did not converge whilst calculating the confidence limits. The returned limits are based on the estimate at the last iteration. Try increasing **Iteration Limit**. |
| 16 | Confidence limits for this value of $\tau$ could not be calculated. The returned upper and lower limits are set to a large positive and large negative value respectively as defined by the optional parameter **Big**. |

It is possible for multiple warnings to be applicable to a single model. In these cases the value returned in **info** is the sum of the corresponding individual nonzero warning codes.

9: **ifail** – INTEGER

**ifail** = 0 unless the function detects an error (see Section 5).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**ifail** = 11

Constraint: **sorder** = 1 or 2.

**ifail** = 21

On entry, **intcpt** = ⟨*value*⟩ was an illegal value.

**ifail** = 31

On entry, **weight** had an illegal value.

**ifail** $= 41$

Constraint: **n** $\geq 2$.

**ifail** $= 51$

Constraint: **m** $\geq 0$.

**ifail** $= 71$

Constraint: $lddat \geq$ **n**.

**ifail** $= 72$

Constraint: $lddat \geq$ **m**.

**ifail** $= 81$

Constraint: **isx**$(i) = 0$ or $1$ for all $i$.

**ifail** $= 91$

Constraint: $1 \leq$ **ip** $<$ **n**.

**ifail** $= 92$

On entry, **ip** is not consistent with **isx** or **intcpt**.

**ifail** $= 111$

Constraint: **wt**$(i) \geq 0.0$ for all $i$.

**ifail** $= 112$

Constraint: effective number of observations $\geq \langle value \rangle$.

**ifail** $= 121$

Constraint: **ntau** $\geq 1$.

**ifail** $= 131$

On entry is invalid.

**ifail** $= 201$

On entry, either the option arrays have not been initialized or they have been corrupted.

**ifail** $= 221$

On entry, **state** vector has been corrupted or not initialized.

**ifail** $= 231$

A potential problem occurred whilst fitting the model(s).
Additional information has been returned in **info**.

**ifail** $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

**ifail** $= -399$

Your licence key may have expired or may not have been installed correctly.

**ifail** $= -999$

Dynamic memory allocation failed.

## 7 Accuracy

Not applicable.

## 8 Further Comments

nag_correg_quantile_linreg (g02qg) allocates internally approximately the following elements of double storage: $13n + np + 3p^2 + 6p + 3(p+1) \times$ **ntau**. If **Interval Method** = BOOTSTRAP XY then a further $np$ elements are required, and this increases by $p \times$ **ntau** $\times$ **Bootstrap Iterations** if **Bootstrap Interval Method** = QUANTILE. Where possible, any user-supplied output arrays are used as workspace and so the amount actually allocated may be less. If **sorder** = 2, **weight** = 'U', **intcpt** = 'N' and **ip** = **m** an internal copy of the input data is avoided and the amount of locally allocated memory is reduced by $np$.

## 9 Example

A quantile regression model is fitted to Engels 1857 study of household expenditure on food. The model regresses the dependent variable, household food expenditure, against two explanatory variables, a column of ones and household income. The model is fit for five different values of $\tau$ and the covariance matrix is estimated assuming Normal IID errors. Both the covariance matrix and the residuals are returned.

### 9.1 Program Text

```
function g02qg_example

fprintf('g02qg example results\n\n');

sorder = nag_int(1);
c1 = 'y';
weight = 'u';
dat = [ 420.1577;  541.4117;  901.1575;  639.0802;  750.8756;  945.7989;
        829.3979;  979.1648; 1309.8789; 1492.3987;  502.8390;  616.7168;
        790.9225;  555.8786;  713.4412;  838.7561;  535.0766;  596.4408;
        924.5619;  487.7583;  692.6397;  997.8770;  506.9995;  654.1587;
        933.9193;  433.6813;  587.5962;  896.4746;  454.4782;  584.9989;
        800.7990;  502.4369;  713.5197;  906.0006;  880.5969;  796.8289;
        854.8791; 1167.3716;  523.8000;  670.7792;  377.0584;  851.5430;
       1121.0937;  625.5179;  805.5377;  558.5812;  884.4005; 1257.4989;
       2051.1789; 1466.3330;  730.0989; 2432.3910;  940.9218; 1177.8547;
       1222.5939; 1519.5811;  687.6638;  953.1192;  953.1192;  953.1192;
        939.0418; 1283.4025; 1511.5789; 1342.5821;  511.7980;  689.7988;
       1532.3074; 1056.0808;  387.3195;  387.3195;  410.9987;  499.7510;
        832.7554;  614.9986;  887.4658; 1595.1611; 1807.9520;  541.2006;
       1057.6767;  800.7990; 1245.6964; 1201.0002;  634.4002;  956.2315;
       1148.6010; 1768.8236; 2822.5330;  922.3548; 2293.1920;  627.4726;
        889.9809; 1162.2000; 1197.0794;  530.7972; 1142.1526; 1088.0039;
        484.6612; 1536.0201;  678.8974;  671.8802;  690.4683;  860.6948;
        873.3095;  894.4598; 1148.6470;  926.8762;  839.0414;  829.4974;
       1264.0043; 1937.9771;  698.8317;  920.4199; 1897.5711;  891.6824;
        889.6784; 1221.4818;  544.5991; 1031.4491; 1462.9497;  830.4353;
        975.0415; 1337.9983;  867.6427;  725.7459;  989.0056; 1525.0005;
        672.1960;  923.3977;  472.3215;  590.7601;  831.7983; 1139.4945;
        507.5169;  576.1972;  696.5991;  650.8180;  949.5802;  497.1193;
        570.1674;  724.7306;  408.3399;  638.6713; 1225.7890;  715.3701;
        800.4708;  975.5974; 1613.7565;  608.5019;  958.6634;  835.9426;
       1024.8177; 1006.4353;  726.0000;  494.4174;  776.5958;  415.4407;
        581.3599;  643.3571; 2551.6615; 1795.3226; 1165.7734;  815.6212;
       1264.2066; 1095.4056;  447.4479; 1178.9742;  975.8023; 1017.8522;
        423.8798;  558.7767;  943.2487; 1348.3002; 2340.6174;  587.1792;
       1540.9741; 1115.8481; 1044.6843; 1389.7929; 2497.7860; 1585.3809;
```

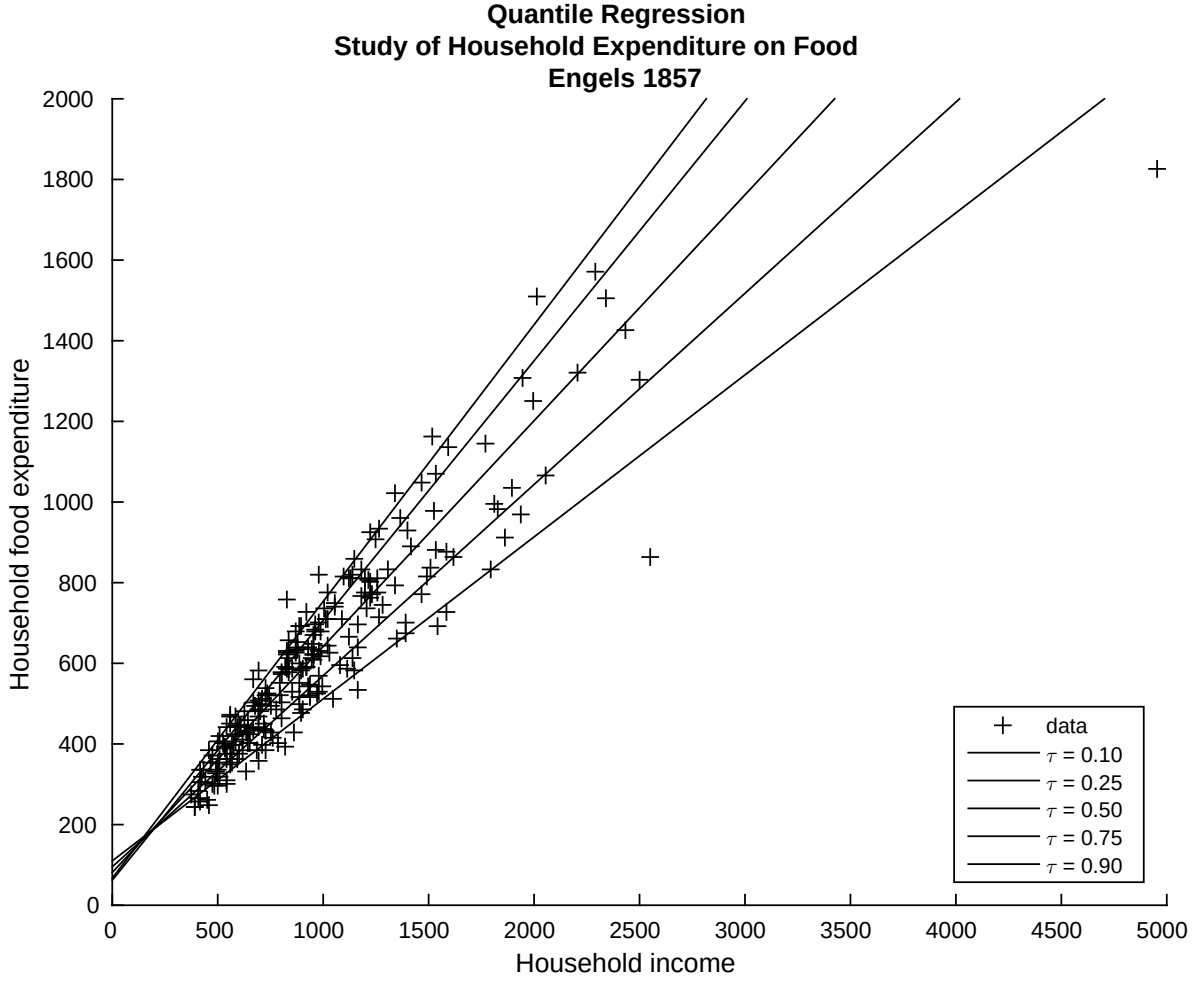
```
        1862.0438; 2008.8546;  697.3099;  571.2517;  598.3465;  461.0977;
         977.1107;  883.9849;  718.3594;  543.8971; 1587.3480; 4957.8130;
         969.6838;  419.9980;  561.9990;  689.5988; 1398.5203;  820.8168;
         875.1716; 1392.4499; 1256.3174; 1362.8590; 1999.2552; 1209.4730;
        1125.0356; 1827.4010; 1014.1540;  880.3944;  873.7375;  951.4432;
         473.0022;  601.0030;  713.9979;  829.2984;  959.7953; 1212.9613;
         958.8743; 1129.4431; 1943.0419;  539.6388;  463.5990;  562.6400;
         736.7584; 1415.4461; 2208.7897;  636.0009;  759.4010; 1078.8382;
         748.6413;  987.6417;  788.0961; 1020.0225; 1230.9235;  440.5174;
         743.0772];
y    = [ 255.8394;  310.9587;  485.6800;  402.9974;  495.5608;  633.7978;
         630.7566;  700.4409;  830.9586;  815.3602;  338.0014;  412.3613;
         520.0006;  452.4015;  512.7201;  658.8395;  392.5995;  443.5586;
         640.1164;  333.8394;  466.9583;  543.3969;  317.7198;  424.3209;
         518.9617;  338.0014;  419.6412;  476.3200;  386.3602;  423.2783;
         503.3572;  354.6389;  497.3182;  588.5195;  654.5971;  550.7274;
         528.3770;  640.4813;  401.3204;  435.9990;  276.5606;  588.3488;
         664.1978;  444.8602;  462.8995;  377.7792;  553.1504;  810.8962;
        1067.9541; 1049.8788;  522.7012; 1424.8047;  517.9196;  830.9586;
         925.5795; 1162.0024;  383.4580;  621.1173;  621.1173;  621.1173;
         548.6002;  745.2353;  837.8005;  795.3402;  418.5976;  508.7974;
         883.2780;  742.5276;  242.3202;  242.3202;  266.0010;  408.4992;
         614.7588;  385.3184;  515.6200; 1138.1620;  993.9630;  299.1993;
         750.3202;  572.0807;  907.3969;  811.5776;  427.7975;  649.9985;
         860.6002; 1143.4211; 2032.6792;  590.6183; 1570.3911;  483.4800;
         600.4804;  696.2021;  774.7962;  390.5984;  612.5619;  708.7622;
         296.9192; 1071.4627;  496.5976;  503.3974;  357.6411;  430.3376;
         624.6990;  582.5413;  580.2215;  543.8807;  588.6372;  627.9999;
         712.1012;  968.3949;  482.5816;  593.1694; 1033.5658;  693.6795;
         693.6795;  761.2791;  361.3981;  628.4522;  771.4486;  757.1187;
         821.5970; 1022.3202;  679.4407;  538.7491;  679.9981;  977.0033;
         561.2015;  728.3997;  372.3186;  361.5210;  620.8006;  819.9964;
         360.8780;  395.7608;  442.0001;  404.0384;  670.7993;  297.5702;
         353.4882;  383.9376;  284.8008;  431.1000;  801.3518;  448.4513;
         577.9111;  570.5210;  865.3205;  444.5578;  680.4198;  576.2779;
         708.4787;  734.2356;  433.0010;  327.4188;  485.5198;  305.4390;
         468.0008;  459.8177;  863.9199;  831.4407;  534.7610;  392.0502;
         934.9752;  813.3081;  263.7100;  769.0838;  630.5863;  645.9874;
         319.5584;  348.4518;  614.5068;  662.0096; 1504.3708;  406.2180;
         692.1689;  588.1371;  511.2609;  700.5600; 1301.1451;  879.0660;
         912.8851; 1509.7812;  484.0605;  399.6703;  444.1001;  248.8101;
         527.8014;  500.6313;  436.8107;  374.7990;  726.3921; 1827.2000;
         523.4911;  334.9998;  473.2009;  581.2029;  929.7540;  591.1974;
         637.5483;  674.9509;  776.7589;  959.5170; 1250.9643;  737.8201;
         810.6772;  983.0009;  708.8968;  633.1200;  631.7982;  608.6419;
         300.9999;  377.9984;  397.0015;  588.5195;  681.7616;  807.3603;
         696.8011;  811.1962; 1305.7201;  442.0001;  353.6013;  468.0008;
         526.7573;  890.2390; 1318.8033;  331.0005;  416.4015;  596.8406;
         429.0399;  619.6408;  400.7990;  775.0209;  772.7611;  306.5191;
         522.6019];

isx = [nag_int(1)];
tau = [0.10; 0.25; 0.50; 0.75; 0.90];
state = zeros(1, 1, nag_int_name);
ip = 2;
b = zeros(2, 5);
iopts = zeros(100, 1, nag_int_name);
opts = zeros(100, 1);

% Initialize the optional argument array
[iopts, opts, ifail] = g02zk( ...
                              'Initialize = g02qg', iopts, opts);

% Set optional arguments
[iopts, opts, ifail] = g02zk( ...
                              'Return Residuals = Yes', iopts, opts);
[iopts, opts, ifail] = g02zk( ...
                              'Matrix Returned = Covariance', iopts, opts);
[iopts, opts, ifail] = g02zk( ...
                              'Interval Method = IID', iopts, opts);
```

```
% Call the model fitting routine
[df, b, bl, bu, ch, res, state, info, ifail] = ...
  g02qg( ...
         sorder, c1, weight, dat, isx, y, tau, b, iopts, opts, state);

% Display the parameter estimates
% plot setup
t = '\tau';
fig1 = figure;
hold on;
plot(dat,y,'+r');
tt{1} = 'data';
% loop over tau
for l=1:numel(tau)
  fprintf('\nQuantile: %6.3f\n\n', tau(l));
  fprintf('        Lower   Parameter   Upper\n');
  fprintf('        Limit   Estimate    Limit\n');
  for j=1:2
    fprintf('%3d   %7.3f   %7.3f   %7.3f\n', j, bl(j,l), b(j,l), bu(j,l));
  end
  fprintf('\nCovariance matrix\n');
  for i=1:ip
    fprintf('%10.3e ', ch(1:i, i, l));
    fprintf('\n');
  end
  fprintf('\n');
  plot([0 (2000-b(1,l))/b(2,l)],[b(1,l) 2000]);
  tt{l+1} = sprintf('%s = %4.2f',t,tau(l));
end
% plot labels
legend(tt,'Location','SouthEast')
xlabel('Household income');
ylabel('Household food expenditure');
title({'Quantile Regression', ...
       ' Study of Household Expenditure on Food', ...
       'Engels 1857'});
axis([0 5000 0 2000]);
hold off;

if (numel(res) > 0)
  fprintf('First 10 Residuals\n');
  fprintf('                                Quantile\n');
  fprintf('Obs.   %6.3f     %6.3f     %6.3f     %6.3f     %6.3f\n', tau);
  for i=1:10
    fprintf(' %3d %10.5f %10.5f %10.5f %10.5f %10.5f\n', i, res(i, 1:5));
  end
else
  fprintf('Residuals not returned\n');
end
```

## 9.2 Program Results

```
     g02qg example results

Quantile:  0.100

        Lower   Parameter   Upper
        Limit   Estimate    Limit
  1    74.946   110.142   145.337
  2     0.370     0.402     0.433

Covariance matrix
 3.191e+02
-2.541e-01  2.587e-04

Quantile:  0.250

        Lower   Parameter   Upper
        Limit   Estimate    Limit
  1    64.232    95.483   126.735
```

```
  2     0.446     0.474     0.502

Covariance matrix
 2.516e+02
-2.004e-01  2.039e-04

Quantile:  0.500

        Lower   Parameter   Upper
        Limit   Estimate    Limit
  1    55.399    81.482   107.566
  2     0.537     0.560     0.584

Covariance matrix
 1.753e+02
-1.396e-01  1.421e-04

Quantile:  0.750

        Lower   Parameter   Upper
        Limit   Estimate    Limit
  1    41.372    62.396    83.421
  2     0.625     0.644     0.663

Covariance matrix
 1.139e+02
-9.068e-02  9.230e-05

Quantile:  0.900

        Lower   Parameter   Upper
        Limit   Estimate    Limit
  1    26.829    67.351   107.873
  2     0.650     0.686     0.723

Covariance matrix
 4.230e+02
-3.369e-01  3.429e-04

First 10 Residuals
                            Quantile
Obs.     0.100      0.250      0.500      0.750      0.900
   1  -23.10718  -38.84219  -61.00711  -77.14462  -99.86551
   2  -16.70358  -41.20981  -73.81193 -100.11463 -127.96277
   3   13.48419  -37.04518 -100.61322 -157.07478 -200.13481
   4   36.09526    4.52393  -36.48522  -70.97584 -102.95390
   5   83.74310   44.08476   -6.54743  -50.41028  -87.11562
   6  143.66660   89.90799   22.49734  -37.70668  -82.65437
   7  187.39134  142.05288   84.66171   34.21603   -5.80963
   8  196.90443  140.73220   70.44951    7.44831  -38.91027
   9  194.55254  114.45726   15.70761  -75.01861 -135.36147
  10  105.62394   12.32563 -102.13482 -208.16238 -276.22311
```




## 10 Algorithmic Details

By the addition of slack variables the minimization (1) can be reformulated into the linear programming problem

$$\underset{(u,v,\beta)\in\mathbb{R}^n_+\times\mathbb{R}^n_+\times\mathbb{R}^p}{\text{minimize}} \tau e^{\mathrm{T}}u + (1-\tau)e^{\mathrm{T}}v \qquad \text{subject to} \qquad y = X\beta + u - v \tag{2}$$

and its associated dual

$$\underset{d}{\text{maximize}} y^{\mathrm{T}}d \qquad \text{subject to} \qquad X^{\mathrm{T}}d = 0, d \in [\tau-1,\tau]^n \tag{3}$$

where $e$ is a vector of $n$ 1s. Setting $a = d + (1-\tau)e$ gives the equivalent formulation

$$\underset{a}{\text{maximize}} y^{\mathrm{T}}a \qquad \text{subject to} \qquad X^{\mathrm{T}}a = (1-\tau)X^{\mathrm{T}}e, a \in [0,1]^n. \tag{4}$$

The algorithm introduced by Portnoy and Koenker (1997) and used by nag_correg_quantile_linreg (g02qg), uses the primal-dual formulation expressed in equations (2) and (4) along with a logarithmic barrier function to obtain estimates for $\beta$. The algorithm is based on the predictor-corrector algorithm of Mehrotra (1992) and further details can be obtained from Portnoy and Koenker (1997) and Koenker (2005). A good description of linear programming, interior point algorithms, barrier functions and Mehrotra's predictor-corrector algorithm can be found in Nocedal and Wright (1999).

## 10.1 Interior Point Algorithm

In this section a brief description of the interior point algorithm used to estimate the model parameters is presented. It should be noted that there are some differences in the equations given here – particularly (7) and (9) – compared to those given in Koenker (2005) and Portnoy and Koenker (1997).

### 10.1.1 Central path

Rather than optimize (4) directly, an additional slack variable $s$ is added and the constraint $a \in [0,1]^n$ is replaced with $a + s = e, a_i \geq 0, s_i \geq 0,$ for $i = 1, 2, \ldots, n$.

The positivity constraint on $a$ and $s$ is handled using the logarithmic barrier function

$$B(a, s, \mu) = y^{\mathrm{T}} a + \mu \sum_{i=1}^{n} (\log a_i + \log s_i).$$

The primal-dual form of the problem is used giving the Lagrangian

$$L(a, s, \beta, u, \mu) = B(a, s, \mu) - \beta^{\mathrm{T}} \big(X^{\mathrm{T}} a - (1-\tau) X^{\mathrm{T}} e\big) - u^{\mathrm{T}}(a + s - e)$$

whose central path is described by the following first order conditions

$$\begin{aligned} X^{\mathrm{T}} a &= (1-\tau) X^{\mathrm{T}} e \\ a + s &= e \\ X\beta + u - v &= y \\ SUe &= \mu e \\ AVe &= \mu e \end{aligned} \tag{5}$$

where $A$ denotes the diagonal matrix with diagonal elements given by $a$, similarly with $S, U$ and $V$. By enforcing the inequalities on $s$ and $a$ strictly, i.e., $a_i > 0$ and $s_i > 0$ for all $i$ we ensure that $A$ and $S$ are positive definite diagonal matrices and hence $A^{-1}$ and $S^{-1}$ exist.

Rather than applying Newton's method to the system of equations given in (5) to obtain the step directions $\delta_\beta, \delta_a, \delta_s, \delta_u$ and $\delta_v$, Mehrotra substituted the steps directly into (5) giving the augmented system of equations

$$\begin{aligned} X^{\mathrm{T}}(a + \delta_a) &= (1-\tau) X^{\mathrm{T}} e \\ (a + \delta_a) + (s + \delta_s) &= e \\ X\big(\beta + \delta_\beta\big) + (u + \delta_u) - (v + \delta_v) &= y \\ (S + \Delta_s)(U + \Delta_u)e &= \mu e \\ (A + \Delta_a)(V + \Delta_v)e &= \mu e \end{aligned} \tag{6}$$

where $\Delta_a, \Delta_s, \Delta_u$ and $\Delta_v$ denote the diagonal matrices with diagonal elements given by $\delta_a, \delta_s, \delta_u$ and $\delta_v$ respectively.

### 10.1.2 Affine scaling step

The affine scaling step is constructed by setting $\mu = 0$ in (5) and applying Newton's method to obtain an intermediate set of step directions

$$\begin{aligned} (X^{\mathrm{T}} W X)\delta_\beta &= X^{\mathrm{T}} W (y - X\beta) + (\tau - 1) X^{\mathrm{T}} e + X^{\mathrm{T}} a \\ \delta_a &= W\big(y - X\beta - X\delta_\beta\big) \\ \delta_s &= -\delta_a \\ \delta_u &= S^{-1} U \delta_a - Ue \\ \delta_v &= A^{-1} V \delta_s - Ve \end{aligned} \tag{7}$$

where $W = (S^{-1}U + A^{-1}V)^{-1}$.

Initial step sizes for the primal ($\hat{\gamma}_P$) and dual ($\hat{\gamma}_D$) parameters are constructed as

$$\begin{aligned} \hat{\gamma}_P &= \sigma \min\left\{ \min_{i, \delta_{a_i} < 0} \{a_i / \delta_{a_i}\}, \min_{i, \delta_{s_i} < 0} \{s_i / \delta_{s_i}\} \right\} \\ \hat{\gamma}_D &= \sigma \min\left\{ \min_{i, \delta_{u_i} < 0} \{u_i / \delta_{u_i}\}, \min_{i, \delta_{v_i} < 0} \{v_i / \delta_{v_i}\} \right\} \end{aligned} \tag{8}$$

where $\sigma$ is a user-supplied scaling factor. If $\hat{\gamma}_P \times \hat{\gamma}_D \geq 1$ then the nonlinearity adjustment, described in Section 11.1.3, is not made and the model parameters are updated using the current step size and directions.

### 10.1.3 Nonlinearity Adjustment

In the nonlinearity adjustment step a new estimate of $\mu$ is obtained by letting

$$\hat{g}\left(\hat{\gamma}_P, \hat{\gamma}_D\right) = (s + \hat{\gamma}_P \delta_s)^{\mathrm{T}}(u + \hat{\gamma}_D \delta_u) + (a + \hat{\gamma}_P \delta_a)^{\mathrm{T}}(v + \hat{\gamma}_D \delta_v)$$

and estimating $\mu$ as

$$\mu = \left(\frac{\hat{g}\left(\hat{\gamma}_P, \hat{\gamma}_D\right)}{\hat{g}\left(0,0\right)}\right)^3 \frac{\hat{g}\left(0,0\right)}{2n}.$$

This estimate, along with the nonlinear terms ($\Delta u$, $\Delta s$, $\Delta a$ and $\Delta v$) from (6) are calculated using the values of $\delta_a, \delta_s, \delta_u$ and $\delta_v$ obtained from the affine scaling step.

Given an updated estimate for $\mu$ and the nonlinear terms the system of equations

$$\begin{aligned}
(X^{\mathrm{T}}WX)\delta_\beta &= X^{\mathrm{T}}W(y - X\beta + \mu(S^{-1} - A^{-1})e + S^{-1}\Delta_s\Delta_u e - A^{-1}\Delta_a\Delta_v e) + (\tau - 1)X^{\mathrm{T}}e + X^{\mathrm{T}}a \\
\delta_a &= W\big(y - X\beta - X\delta_\beta + \mu(S^{-1} - A^{-1})\big) \\
\delta_s &= -\delta_a \\
\delta_u &= \mu S^{-1}e + S^{-1}U\delta_a - Ue - S^{-1}\Delta_s\Delta_u e \\
\delta_v &= \mu A^{-1}e + A^{-1}V\delta_s - Ve - A^{-1}\Delta_a\Delta_v e
\end{aligned} \tag{9}$$

are solved and updated values for $\delta_\beta, \delta_a, \delta_s, \delta_u, \delta_v, \hat{\gamma}_P$ and $\hat{\gamma}_D$ calculated.

### 10.1.4 Update and convergence

At each iteration the model parameters $(\beta, a, s, u, v)$ are updated using step directions, $\left(\delta_\beta, \delta_a, \delta_s, \delta_u, \delta_v\right)$ and step lengths $(\hat{\gamma}_P, \hat{\gamma}_D)$.

Convergence is assessed using the duality gap, that is, the differences between the objective function in the primal and dual formulations. For any feasible point $(u, v, s, a)$ the duality gap can be calculated from equations (2) and (3) as

$$\begin{aligned}
\tau e^{\mathrm{T}}u + (1-\tau)e^{\mathrm{T}}v - d^{\mathrm{T}}y &= \tau e^{\mathrm{T}}u + (1-\tau)e^{\mathrm{T}}v - (a - (1-\tau)e)^{\mathrm{T}}y \\
&= s^{\mathrm{T}}u + a^{\mathrm{T}}v \\
&= e^{\mathrm{T}}u - a^{\mathrm{T}}y + (1-\tau)e^{\mathrm{T}}X\beta
\end{aligned}$$

and the optimization terminates if the duality gap is smaller than the tolerance supplied in the optional parameter **Tolerance**.

### 10.1.5 Additional information

Initial values are required for the parameters $a, s, u, v$ and $\beta$. If not supplied by the user, initial values for $\beta$ are calculated from a least squares regression of $y$ on $X$. This regression is carried out by first constructing the cross-product matrix $X^{\mathrm{T}}X$ and then using a pivoted $QR$ decomposition as performed by nag_lapack_dgeqp3 (f08bf). In addition, if the cross-product matrix is not of full rank, a rank reduction is carried out and, rather than using the full design matrix, $X$, a matrix formed from the first $p$-rank columns of $XP$ is used instead, where $P$ is the pivot matrix used during the $QR$ decomposition. Parameter estimates, confidence intervals and the rows and columns of the matrices returned in the argument **ch** (if any) are set to zero for variables dropped during the rank-reduction. The rank reduction step is performed irrespective of whether initial values are supplied by the user.

Once initial values have been obtained for $\beta$, the initial values for $u$ and $v$ are calculated from the residuals. If $|r_i| < \epsilon_u$ then a value of $\pm\epsilon_u$ is used instead, where $\epsilon_u$ is supplied in the optional parameter **Epsilon**. The initial values for the $a$ and $s$ are always set to $1 - \tau$ and $\tau$ respectively.

The solution for $\delta_\beta$ in both (7) and (9) is obtained using a Bunch–Kaufman decomposition, as implemented in nag_lapack_dsytrf (f07md).

### 10.2 Calculation of Covariance Matrix

nag_correg_quantile_linreg (g02qg) supplies four methods to calculate the covariance matrices associated with the parameter estimates for $\beta$. This section gives some additional detail on three of the algorithms, the fourth, (which uses bootstrapping), is described in Section 3.

(i) Independent, identically distributed (IID) errors

When assuming IID errors, the covariance matrices depend on the sparsity, $s(\tau)$, which nag_correg_quantile_linreg (g02qg) estimates as follows:

(a) Let $r_i$ denote the residuals from the original quantile regression, that is $r_i = y_i - x_i^{\mathrm{T}}\hat{\beta}$.

(b) Drop any residual where $|r_i|$ is less than $\epsilon_u$, supplied in the optional parameter **Epsilon**.

(c) Sort and relabel the remaining residuals in ascending order, by absolute value, so that $\epsilon_u < |r_1| < |r_2| < \ldots$.

(d) Select the first $l$ values where $l = h_n n$, for some bandwidth $h_n$.

(e) Sort and relabel these $l$ residuals again, so that $r_1 < r_2 < \ldots < r_l$ and regress them against a design matrix with two columns ($p = 2$) and rows given by $x_i = \{1, i/(n-p)\}$ using quantile regression with $\tau = 0.5$.

(f) Use the resulting estimate of the slope as an estimate of the sparsity.

(ii) Powell Sandwich

When using the Powell Sandwich to estimate the matrix $H_n$, the quantity

$$c_n = \min(\sigma_r, (q_{r3} - q_{r1})/1.34) \times \left(\Phi^{-1}(\tau + h_n) - \Phi^{-1}(\tau - h_n)\right)$$

is calculated. Dependent on the value of $\tau$ and the method used to calculate the bandwidth ($h_n$), it is possible for the quantities $\tau \pm h_n$ to be too large or small, compared to ***machine precision*** ($\epsilon$). More specifically, when $\tau - h_n \leq \sqrt{\epsilon}$, or $\tau + h_n \geq 1 - \sqrt{\epsilon}$, a warning flag is raised in **info**, the value is truncated to $\sqrt{\epsilon}$ or $1 - \sqrt{\epsilon}$ respectively and the covariance matrix calculated as usual.

(iii) Hendricks–Koenker Sandwich

The Hendricks–Koenker Sandwich requires the calculation of the quantity $d_i = x_i^{\mathrm{T}}\left(\hat{\beta}(\tau + h_n) - \hat{\beta}(\tau - h_n)\right)$. As with the Powell Sandwich, in cases where $\tau - h_n \leq \sqrt{\epsilon}$, or $\tau + h_n \geq 1 - \sqrt{\epsilon}$, a warning flag is raised in **info**, the value truncated to $\sqrt{\epsilon}$ or $1 - \sqrt{\epsilon}$ respectively and the covariance matrix calculated as usual.

In addition, it is required that $d_i > 0$, in this method. Hence, instead of using $2h_n/d_i$ in the calculation of $H_n$, $\max(2h_n/(d_i + \epsilon_u), 0)$ is used instead, where $\epsilon_u$ is supplied in the optional parameter **Epsilon**.

## 11 Optional Parameters

Several optional parameters in nag_correg_quantile_linreg (g02qg) control aspects of the optimization algorithm, methodology used, logic or output. Their values are contained in the arrays **iopts** and **opts**; these must be initialized before calling nag_correg_quantile_linreg (g02qg) by first calling nag_correg_optset (g02zk) with **optstr** set to **Initialize** = g02qg.

Each optional parameter has an associated default value; to set any of them to a non-default value, use nag_correg_optset (g02zk). The current value of an optional parameter can be queried using nag_correg_optget (g02zl).

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 12.1.

**Band Width Alpha**

**Band Width Method**

**Big**
**Bootstrap Interval Method**
**Bootstrap Iterations**
**Bootstrap Monitoring**
**Calculate Initial Values**
**Defaults**
**Drop Zero Weights**
**Epsilon**
**Interval Method**
**Iteration Limit**
**Matrix Returned**
**Monitoring**
**QR Tolerance**
**Return Residuals**
**Sigma**
**Significance Level**
**Tolerance**
**Unit Number**

## 11.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

the keywords, where the minimum abbreviation of each keyword is underlined (if no characters of an optional qualifier are underlined, the qualifier may be omitted);

a parameter value, where the letters $a$, $i$ and $r$ denote options that take character, integer and real values respectively;

the default value, where the symbol $\epsilon$ is a generic notation for ***machine precision*** (see nag_machine_precision (x02aj)).

Keywords and character values are case and white space insensitive.

**Band Width Alpha** $r$ Default $= 1.0$

A multiplier used to construct the parameter $\alpha_b$ used when calculating the Sheather–Hall bandwidth (see Section 3), with $\alpha_b = (1 - \alpha) \times$ **Band Width Alpha**. Here, $\alpha$ is the **Significance Level**.

*Constraint*: **Band Width Alpha** $> 0.0$.

**Band Width Method** $a$ Default $=$ 'SHEATHER HALL'

The method used to calculate the bandwidth used in the calculation of the asymptotic covariance matrix $\Sigma$ and $H^{-1}$ if **Interval Method** $=$ HKS, KERNEL or IID (see Section 3).

*Constraint*: **Band Width Method** $=$ SHEATHER HALL or BOFINGER.

**Big** $r$ Default $= 10.0^{20}$

This parameter should be set to something larger than the biggest value supplied in **dat** and **y**.

*Constraint*: **Big** $> 0.0$.

**Bootstrap Interval Method** $a$ Default $=$ QUANTILE

If **Interval Method** $=$ BOOTSTRAP XY, **Bootstrap Interval Method** controls how the confidence intervals are calculated from the bootstrap estimates.

**Bootstrap Interval Method** $=$ T

$t$ intervals are calculated. That is, the covariance matrix, $\Sigma = \left\{\sigma_{ij} : i, j = 1, 2, \ldots, p\right\}$ is calculated from the bootstrap estimates and the limits calculated as $\beta_i \pm t_{(n-p,(1+\alpha)/2)}\sigma_{ii}$ where $t_{(n-p,(1+\alpha)/2)}$ is the $(1+\alpha)/2$ percentage point from a Student's $t$ distribution on $n-p$ degrees of freedom, $n$ is the effective number of observations and $\alpha$ is given by the optional parameter **Significance Level**.

**Bootstrap Interval Method** $=$ QUANTILE

Quantile intervals are calculated. That is, the upper and lower limits are taken as the $(1+\alpha)/2$ and $(1-\alpha)/2$ quantiles of the bootstrap estimates, as calculated using nag_stat_quantiles (g01am).

*Constraint*: **Bootstrap Interval Method** $=$ T or QUANTILE.

**Bootstrap Iterations** $i$ Default $= 100$

The number of bootstrap samples used to calculate the confidence limits and covariance matrix (if requested) when **Interval Method** $=$ BOOTSTRAP XY.

*Constraint*: **Bootstrap Iterations** $> 1$.

**Bootstrap Monitoring** $a$ Default $=$ NO

If **Bootstrap Monitoring** $=$ YES and **Interval Method** $=$ BOOTSTRAP XY, then the parameter estimates for each of the bootstrap samples are displayed. This information is sent to the unit number specified by **Unit Number**.

*Constraint*: **Bootstrap Monitoring** $=$ YES or NO.

**Calculate Initial Values** $a$ Default $=$ YES

If **Calculate Initial Values** $=$ YES then the initial values for the regression parameters, $\beta$, are calculated from the data. Otherwise they must be supplied in **b**.

*Constraint*: **Calculate Initial Values** $=$ YES or NO.

**Defaults**

This special keyword is used to reset all optional parameters to their default values.

**Drop Zero Weights** $a$ Default $=$ YES

If a weighted regression is being performed and **Drop Zero Weights** $=$ YES then observations with zero weight are dropped from the analysis. Otherwise such observations are included.

*Constraint*: **Drop Zero Weights** $=$ YES or NO.

**Epsilon** $r$ Default $= \sqrt{\epsilon}$

$\epsilon_u$, the tolerance used when calculating the covariance matrix and the initial values for $u$ and $v$. For additional details see Section 11.2 and Section 11.1.5 respectively.

*Constraint*: **Epsilon** $\geq 0.0$.

**Interval Method** $a$ Default $=$ IID

The value of **Interval Method** controls whether confidence limits are returned in **bl** and **bu** and how these limits are calculated. This parameter also controls how the matrices returned in **ch** are calculated.

**Interval Method** $=$ NONE

No limits are calculated and **bl**, **bu** and **ch** are not referenced.

**Interval Method** = KERNEL
The Powell Sandwich method with a Gaussian kernel is used.

**Interval Method** = HKS
The Hendricks–Koenker Sandwich is used.

**Interval Method** = IID
The errors are assumed to be identical, and independently distributed.

**Interval Method** = BOOTSTRAP XY
A bootstrap method is used, where sampling is done on the pair $(y_i, x_i)$. The number of bootstrap samples is controlled by the parameter **Bootstrap Iterations** and the type of interval constructed from the bootstrap samples is controlled by **Bootstrap Interval Method**.

*Constraint*: **Interval Method** = NONE, KERNEL, HKS, IID or BOOTSTRAP XY.

**Iteration Limit** $i$ Default = 100

The maximum number of iterations to be performed by the interior point optimization algorithm.

*Constraint*: **Iteration Limit** $> 0$.

**Matrix Returned** $a$ Default = NONE

The value of **Matrix Returned** controls the type of matrices returned in **ch**. If **Interval Method** = NONE, this parameter is ignored and **ch** is not referenced. Otherwise:

**Matrix Returned** = NONE
No matrices are returned and **ch** is not referenced.

**Matrix Returned** = COVARIANCE
The covariance matrices are returned.

**Matrix Returned** = H INVERSE
If **Interval Method** = KERNEL or HKS, the matrices $J$ and $H^{-1}$ are returned. Otherwise no matrices are returned and **ch** is not referenced.

The matrices returned are calculated as described in Section 3, with the algorithm used specified by **Interval Method**. In the case of **Interval Method** = BOOTSTRAP XY the covariance matrix is calculated directly from the bootstrap estimates.

*Constraint*: **Matrix Returned** = NONE, COVARIANCE or H INVERSE.

**Monitoring** $a$ Default = NO

If **Monitoring** = YES then the duality gap is displayed at each iteration of the interior point optimization algorithm. In addition, the final estimates for $\beta$ are also displayed.

The monitoring information is sent to the unit number specified by **Unit Number**.

*Constraint*: **Monitoring** = YES or NO.

**QR Tolerance** $r$ Default = $\epsilon^{0.9}$

The tolerance used to calculate the rank, $k$, of the $p \times p$ cross-product matrix, $X^{\mathrm{T}}X$. Letting $Q$ be the orthogonal matrix obtained from a $QR$ decomposition of $X^{\mathrm{T}}X$, then the rank is calculated by comparing $Q_{ii}$ with $Q_{11} \times$ **QR Tolerance**.

If the cross-product matrix is rank deficient, then the parameter estimates for the $p - k$ columns with the smallest values of $Q_{ii}$ are set to zero, along with the corresponding entries in **bl**, **bu** and **ch**, if returned. This is equivalent to dropping these variables from the model. Details on the $QR$ decomposition used can be found in nag_lapack_dgeqp3 (f08bf).

*Constraint*: **QR Tolerance** $> 0.0$.

**Return Residuals** $a$ Default $=$ NO

If **Return Residuals** $=$ YES, the residuals are returned in **res**. Otherwise **res** is not referenced.

*Constraint*: **Return Residuals** $=$ YES or NO.

**Sigma** $r$ Default $= 0.99995$

The scaling factor used when calculating the affine scaling step size (see equation (8)).

*Constraint*: $0.0 <$ **Sigma** $< 1.0$.

**Significance Level** $r$ Default $= 0.95$

$\alpha$, the size of the confidence interval whose limits are returned in **bl** and **bu**.

*Constraint*: $0.0 <$ **Significance Level** $< 1.0$.

**Tolerance** $r$ Default $= \sqrt{\epsilon}$

Convergence tolerance. The optimization is deemed to have converged if the duality gap is less than **Tolerance** (see Section 11.1.4).

*Constraint*: **Tolerance** $> 0.0$.

**Unit Number** $i$ Default taken from nag_file_set_unit_advisory (x04ab)

The unit number to which any monitoring information is sent.

*Constraint*: **Unit Number** $> 1$.

## 12 Description of Monitoring Information

See the description of the optional argument **Monitoring**.